University of Glasgow

DEPARTMENT OF

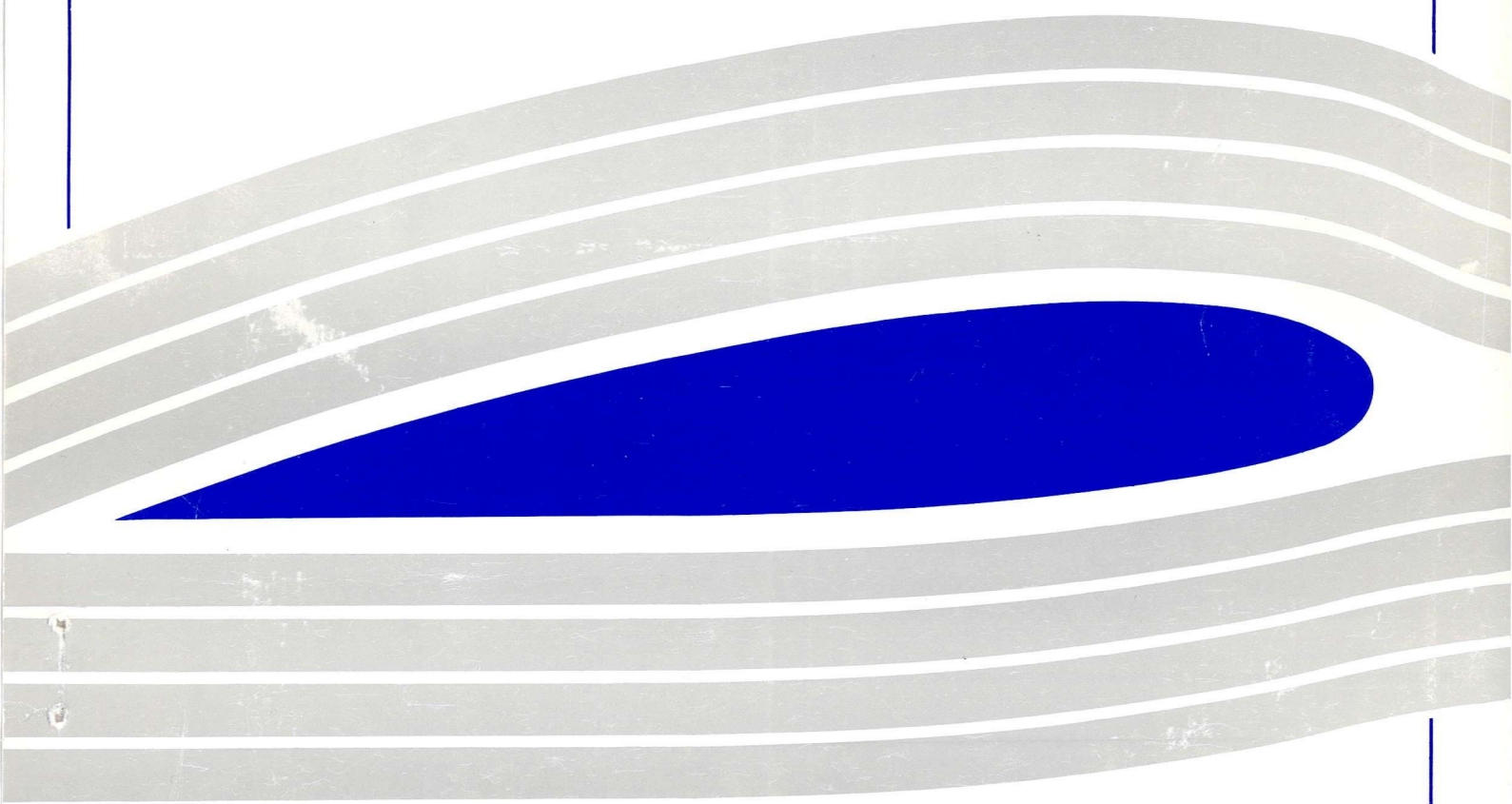
AEROSPACE ENGINEERING

Newton's Method for Steady Laminar Aerofoil Flows

K.J. Badcock

Aero Report 9310

April 29, 1993



Newton's Method for Steady Laminar Aerofoil Flows

K.J. Badcock

Aero Report 9310

April 29, 1993

ENGINEER
PEE
J5000

Newton's Method for Steady Laminar Aerofoil Flows

K.J. Badcock *†
Aerospace Engineering Department
University of Glasgow,
Glasgow, G12 8QQ, U.K.
EMAIL:gnaa36@uk.ac.glasgow.udcf

April 29, 1993

Abstract

Newton's method for steady laminar flows over aerofoils is examined as a prelude to the study of more demanding unsteady flows. A simple and fast method for approximating the Jacobian is proposed and various conjugate gradient techniques are evaluated for the solution of the linear system. Rapid convergence is demonstrated on two test cases.

*This work was carried out under SERC contract.

†The author would like to thank Prof B.E.Richards and Mr X.Xu of the University of Glasgow and Dr N.Qin of the University of Coventry for helpful discussions on Newton's method. He would also like to acknowledge the help of Dr P.J. Wesson and Mr I.C. Glover of the Oxford University Computing Laboratory with Conjugate Gradient methods and with the computations respectively.

Contents

1 Introduction

The quadratic convergence of Newton's method makes it attractive for the solution of the nonlinear algebraic equations which arise from implicit discretisations in computational fluid dynamics (CFD). In this report we examine a Newton-type method originally developed for hypersonic cone flows in [1] [2] [3] with reference to aerofoil flows. Below, a brief survey of the applications of Newton's method to Euler and Navier-Stoke's calculations is given.

A variety of techniques relating to the calculation of the Jacobian and the solution of the linear system were developed in [1] [2] [3]. For generality, the Jacobian was calculated by a finite-difference approximation. The step size can be chosen to retain quadratic convergence. If proper account is taken of which residuals depend on which unknowns [4] [1] then the number of function evaluations can be limited to 5x5x4 for the stencils used in the present work. The problem of solving the linear was tackled in [3] where GMRES was used together with a preconditioning strategy which was designed to be easily parallelisable. A block diagonal preconditioner was employed together with a diagonal damping factor to obtain fast convergence of the GMRES solution. An outer iterative loop was used to regain the solution to the original system. These techniques were all applied to a hypersonic cone test problem and fast convergence was noted when compared to a local time-stepping method.

Newton's method was applied to the Navier-Stokes equations for aerofoil flows in [5] [6] . The scheme used an implicit time-stepping procedure equivalent to Newton's method with a damping factor inversely proportional to the time step and the analytical Jacobian was calculated. A sparse, direct solver was applied and mesh sequencing was found to speed up the rate of convergence with coarse grid solutions being used to provide initial solutions on finer grids. Tests were performed on flows over NACA 0012 aerofoils and the method was found to have the robustness of explicit methods whilst retaining fast convergence once a sufficiently good solution had been reached.

In [7] the use of the iterative solver GMRES along with incomplete LU decomposition (ILU) preconditioning was investigated with the schemes of [5] [6] . A Baldwin-Lomax turbulence model was used although its contribution to the Jacobian was neglected. The ILU preconditioner was found to work well for test cases consisting of inviscid transonic flow over an aerofoil, laminar viscous subsonic flow over a NACA 0012 aerofoil at a Reynold's number of 5000 and transonic turbulent flow over an RAE2822 aerofoil at a Reynolds number of 6.5×10^6.

The calculation of the Jacobian represents an obstacle to the successful application of Newton's method. As remarked above, an efficient finite difference calculation is used in this paper. A similar approach was used in [8] and results were obtained for transonic compressible flow over an Onera M6 wing and for incompressible flow around a sphere, over a flat plate and around a ship's hull. A direct way of tackling the complexity of the Jacobian calculation was used in [9] where the symbolic manipulation package MACSYMA was used to calculate analytical expressions for the derivatives which were then output directly to FORTRAN code. The method was tested on flat plate and wedge flows.

An interesting application of Newton's method has been to the study of non-uniqueness for the solutions of the nonlinear algebraic discrete equations arising from CFD. In [10] a number of solutions were obtained for the Euler and Navier-Stokes equations and their stability was examined. The problems studied were inviscid and viscous flow in a nozzle, flows over a NACA 0012 aerofoil and flows around a cylinder. The linear systems obtained were all small enough to allow direct solution although the comment is made that sparse matrix solvers should be more widely utilised in CFD. Similar non-uniqueness effects were noted in [11] for solutions of the characteristic form of the Euler equations for nozzle flows.

In this report Newton's method is considered as a means of quickly solving the nonlinear discrete equations once a suitable starting solution has been obtained. This is different in philosophy from the approaches of [7] [9] [8] where the contribution from the time derivative makes the method robust from the point of view of choosing a starting solution but also compromises the quadratic convergence obtained by the straight Newton's method. In section 2 a brief formulation of the discretisation is given, the calculation of the Jacobian is discussed in section 3 and the solution of the linear system is examined in section 4. Results for two laminar test problems are given in section 5.

2 Explicit and Implicit Discretisation

The two-dimensional laminar Navier-Stokes equations are given in Cartesian coordinates by

$$\frac{\partial w}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = \frac{\partial R}{\partial x} + \frac{\partial S}{\partial y} \tag{1}$$

where

$$w = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{bmatrix}, G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{bmatrix}$$

$$R = \begin{bmatrix} 0 \\ \sigma_{xx} \\ \sigma_{xy} \\ u\sigma_{xx} + v\sigma_{xy} - q_x \end{bmatrix}, S = \begin{bmatrix} 0 \\ \sigma_{xy} \\ \sigma_{yy} \\ u\sigma_{xy} + v\sigma_{yy} - q_y \end{bmatrix}.$$

Here,

$$\sigma_{xx} = 2\mu u_x - \frac{2}{3}\mu(u_x + v_y), \sigma_{yy} = 2\mu v_y - \frac{2}{3}\mu(u_x + v_y),$$

$$\sigma_{xy} = \sigma_{yx} = \mu(u_y + v_x),$$

$$q_x = -\kappa\frac{\partial T}{\partial x}, q_y = -\kappa\frac{\partial T}{\partial y}, p = (\gamma - 1)(e - \frac{1}{2}\rho(u^2 + v^2)),$$

$$T = c_v(\frac{e}{\rho} - \frac{1}{2}(u^2 + v^2)).$$

The symbols ρ, u, v, e, p, μ, κ, T represent the fluid density, the two components of velocity, energy, pressure, viscosity, heat conductivity and temperature respectively. The constants γ and c_v stand for the ratio of the specific heats and the specific heat at constant volume respectively. The viscosity is assumed to vary with temperature by Sutherland's law. In this paper the thin-layer form of the non-dimensionalised equations are solved in general curvilinear coordinates. The non-dimensionalisation is with respect to the freestream temperature, Mach number and velocity.

To summarise the solution procedure, the convective terms are discretised by Osher's approximate Riemann Solver [12] with a MUSCL interpolation [13] providing 2nd or 3rd order accuracy. When a limiter is required in the following, the Van Albada limiter [14] is used. The viscous terms are centrally differenced and Riemann invariants are used to impose far field boundary conditions. The explicit method can be written in the form

$$w^{n+1} = w^n - \frac{\Delta t}{\Delta x} R_s(w^n) \tag{2}$$

where R_s denotes the steady residual obtained from the discretisation.

Newton's method for solving the steady equations is given by

$$\frac{\partial R_s}{\partial w}(w^{n+1} - w^n) = -R_s(w^n). \tag{3}$$

and the local time stepping form of 2 is used to give a starting solution. The question of what constitutes a good enough starting solution is answered empirically and varies from flow to flow. However, as a general rule the residual of the steady equations must usually be reduced by two orders from the free stream residual before Newton's method converges. If convergence is achieved then the residual falls rapidly with typically less than four steps required to obtain an accurate approximation to the steady discrete equations.

For an efficient method there are two important issues that must be addressed. First, the Jacobian of the discretisation is required. Osher's flux function is differentiable and so represents a suitable choice for the inviscid approximation. This is not the case with Roe's flux function but published results indicate that the lack of differentiability is not severe enough to effect the convergence. The complicated nature of the functions means that an analytical form for the Jacobian is very cumbersome to calculate and encode by hand in a computer program. However, the analytical expressions have been used for Roe's flux function [5] and an efficient code development procedure using symbolic manipulation to calculate the derivatives together with a code optimiser/generator was introduced in [9]. For Osher's scheme, the use of this method would still be awkward due to the large number of conditionals present in the algorithm. The use of a finite difference approximation overcomes the problems of complexity in calculating the Jacobian but can be expensive. In the present work an efficient method of approximating the Jacobian by finite-differences is used and the details are given below in section 3

The second main problem associated with Newton's method is the need to solve a large system of linear equations at each step. The system in the present case is sparse and both direct solvers [5] [6]and iterative solvers [7] have been used in the past. In this paper iterative conjugate gradient type methods are used which are preconditioned by an incomplete LU decomposition. A comparison is made between four methods and different preconditioners in section 4.

3 Approximation of the Jacobian

As discussed above the use of a finite difference approximation to the Jacobian of the discretisation is attractive because of the simplicity of the calculation and the implementation. However, the process is computationally intensive and so efficient methods must be used if the resulting computer code is not to be quick to develop but too slow to use.

In [4] the problem of computing a finite difference approximation to a sparse Jacobian is addressed. To fix ideas assume that the function whose Jacobian A we wish to approximate is denoted by R. The following algorithm can be used to minimise the number of function evaluations required:

- select a set of the indices S of the components of w such that $A_{i,j} = 0 \forall i, j \in S,\ i \neq j$
- perturb $w_i \forall i \in S$

- evaluate R for the perturbed argument and calculate the finite difference approximation to the terms $A_{i,j}$ for $1 < i < n$ and $\forall j \in S$ where n is the dimension of w.
- repeat the last two steps for a different S until all of the components of A have been evaluated.

Note that in each cell there are four unknowns, only one of which may be perturbed for each function evaluation. The total number of evaluations required is $3 \times 3 \times 4$ for the first order method and is $5 \times 5 \times 4$ for the second order method. It can be seen that the evaluation is indeed expensive. This method considers the cell-based residuals and was used in [2].

However, a more efficient procedure arises from decomposing the residual flux by flux. We shall consider the numerical fluxes $g_{i,j+1/2}$ and $v_{i,j+1/2}$. The method for $f_{i+1/2,j}$ is similar to that used for $g_{i,j+1/2}$.

The viscous flux $v_{i,j+1/2}$ depends on the values in cells (i,j) and (i,j+1). It is fairly straightforward to calculate and encode the contributions to the Jacobian due to the relatively simple central difference approximations used.

For the convective terms the complicated nature of the numerical fluxes arising from Osher's scheme makes analytical expressions unattractive. If MUSCL interpolation is used then the flux $g_{i,j+1/2}$ depends on values in the cells (i,j-1), (i,j), (i,j+1) and (i,j+2). We can denote this by

$$g_{i,j+1/2} = g_{i,j+1/2}(w_{i,j-1}, w_{i,j}, w_{i,j+1}, w_{i,j+2}) \tag{4}$$

which in turn can be rewritten as

$$g_{i,j+1/2} = g_{i,j+1/2}(w_l, w_r) \tag{5}$$

where

$$w_l = w_l(w_{i,j-1}w_{i,j}, w_{i,j+1})$$

and

$$w_r = w_r(w_{i,j}, w_{i,j+1}, w_{i,j+2}).$$

To calculate the derivatives of $g_{i,j+1/2}$ the chain rule is used to yield

$$\frac{\partial g_{i,j+1/2}}{w_{i,k}} = \frac{\partial g_{i,j+1/2}}{w_l}\frac{\partial w_l}{w_{i,k}} + \frac{\partial g_{i,j+1/2}}{w_r}\frac{\partial w_r}{w_{i,k}}. \tag{6}$$

The derivatives of the MUSCL interpolation can be evaluated analytically and the symbolic algebra package *REDUCE* has been used to quickly generate efficient code. The *REDUCE* code and the resulting *FORTRAN* code are shown below in appendix A and they illustrate the advantage of symbolic manipulation in code development. The calculation of the terms

mesh	Residual-based	Flux-based	speed up
64 × 16	33.4	3.3	10.1
128 × 32	137.6	15.5	8.8

Table 1: *Comparison of CPU times in seconds for the residual-based and flux-based methods for calculating the Jacobian on different meshes.*

in $\partial g_{i,j+1/2}/\partial w_l$ by finite differences requires four evaluations of $g_{i,j+1/2}$ and similarly for $\partial g_{i,j+1}/\partial w_r$, leading to eight evaluations in total.

The derivatives for each flux can be added into the Jacobian as they are calculated. The flux-based method requires an equivalent of eight function evaluations instead of the one-hundred required for the cell-based method. In theory the new method should be over twelve times faster. This neglects the computation of the MUSCL derivatives which turns out to be fast compared to one function evaluation. The actual comparison for several mesh sizes is given in 1. The theoretical speed up is not observed due to the cost of additional multiplications in composing the chain rule and searching for the correct place in the matrix to place the derivative approximations. However, the speed up is still by a factor of around ten.

4 The Linear Solver

The use of an implicit method invariably gives rise to a large sparse system of linear equations. This is true whether Newton's method is being used to solve the exact discrete nonlinear system of equations very accurately or if a linearisation is used as part of the discrete scheme. In this section various iterative solvers and preconditioners are examined so that the substantial work required to solve the linear system arising from the finite difference Newton's method can be minimised.

4.1 Sparse Data Structure and the matrix generation

The data structure used in the present study was chosen for its utility in programming iterative solvers and preconditioners and its use allowed several existing subroutines to be used.

The matrix information is stored in six vectors with the first three refering to the part of the matrix above the diagonal and the last three to the lower part. For the upper part, the matrix values are stored in a vector of reals with their columns being stored in a corresponding vector

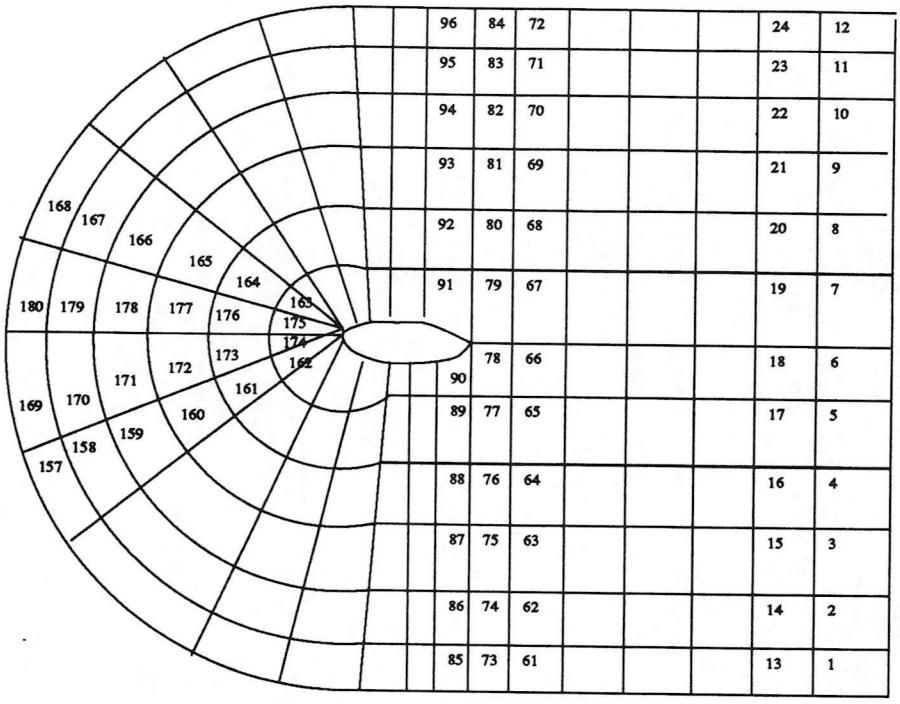


Figure 1: *Ordering of the cells for the linear system.*

of integers. The starting point in these vectors for each row is stored in a third vector of integers. The storage for the lower part of the matrix is similar except that the role of the rows and columns is interchanged.

The generation of the matrix values requires a mapping between the cells in the finite volume discretisation and the unknowns in the linear system. It is important to minimise the band width in order to limit the fill-in required for an effective preconditioner. Following the ordering in previous hypersonic cone flows, one possibility is to number the cells from the lower downstream boundary around the aerofoil to the upper downstream boundary. The resulting band width is $4 \times 3 \times In$ where In is the number of cells around the cone. An alternative ordering is also shown in figure 1 and has a band width of $4 \times 3 \times Jn$. For the cone flows, where typically In and Jn are of a similar size there is nothing to choose between the two and the first ordering was the one used. However, in the present case In is larger than Jn and so the second ordering is used.

The pointer structure for the matrix is the same for all Newton steps and so is generated once at the start of the simulation. As matrix elements are calculated they are substituted into the vector of values after a search through the column or row pointers in the appropriate row or column to find the exact position.

4.2 Iterative Solvers and Preconditioners

The iterative solvers tested were the conjugate gradient type methods for unsymmetric systems called BICG [15], CGSTAB [16], CGS [17] and GMRES [18]. A preconditioner is usually essential for the successful use of these methods. The general idea is to replace the problem

$$Ax = b \tag{7}$$

with the problem

$$CAx = Cb \tag{8}$$

where $C \approx A^{-1}$. The better C approximates A^{-1}, the closer the preconditioned matrix is to the identity matrix.

One popular way of generating C is to approximately factorise A into its upper U and lower L parts and to limit the entries with reference to the sparsity pattern of A. Non-zero entries in L and U that are zero in A are termed fill-in. The case where no fill-in is allowed is called ILU0 where the LU decomposition is defined by

$$(LU)_{ij} = \begin{cases} 0 \text{ if } A_{ij} = 0 \\ A_{ij} \text{ if } A_{i,j} \neq 0 \end{cases} \tag{9}$$

The higher the level of fill-in allowed, the closer the approximate factorisation is to being exact at the cost of the greater storage requirements for L and U and the greater computation required for forming matrix-vector products in the conjugate gradient solution. An algorithm for defining increasing levels of fill-in was given in [19].

Another way of approximating the inverse of A is to find the exact inverse of a matrix consisting of a limited number of diagonals of A. In the present case the diagonal blocks of A represents a convenient approximation to A and this is easily inverted to yield C. However, this doesn't prove to be an effective preconditioner and in [3] a positive number was added to the diagonal to obtain convergence of the iterative solver and an outer iteration was used to regain the solution of the original linear system.

4.3 Comparison of the Conjugate Gradient Solvers

The four iterative solvers were tested on several systems arising from Newton's method The plot of the L_2 norm of the conjugate gradient update against CPU time for the four methods is shown in figures 2 - 5 for a system arising from one of the test cases of the next section. It can be seen

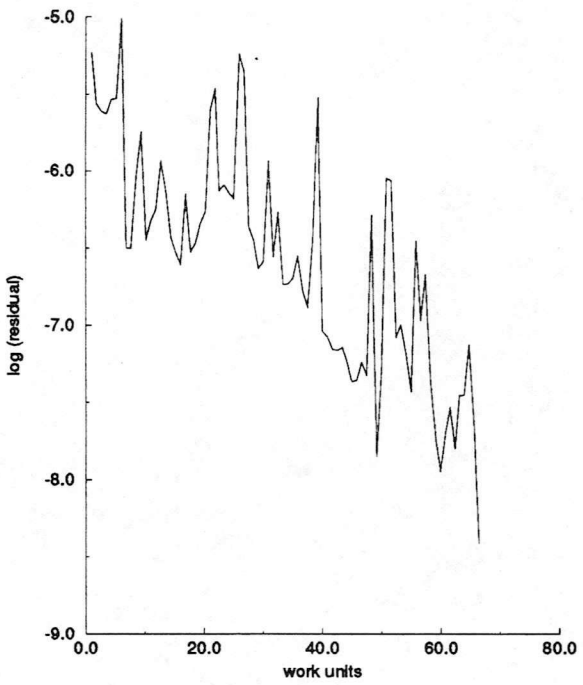


Figure 2: *The convergence for BICG with ILU0 for a linear system arising from Newton's method. The convergence criterion used is for the discrete L_2 norm of the cg update to be reduced below 10^{-8} and a maximum number of iterations is set for each method. The matrix arises from the flow over a NACA0012 aerofoil with Mach number 0.5, Reynolds number 5000 and zero incidence.*

that no solver has a major advantage for this case when the solution times are compared with the preconditioning time of around 500 work units[1]. The GMRES method reduces the residual more smoothly than the other three methods. For all of the cases examined no single method was seen to have a major influence on the convergence of the flow solution.

4.4 Comparison of Preconditioners

The crucial element for solving the linear systems arising from Newton's method by conjugate gradient algorithms is the preconditioner. The balance for ILU preconditioners is between accurate LU decompositions which reduce the number of conjugate gradient steps required for convergence but which increase the cost of each step, are costly to compute and require more storage and more approximate LU decompositions which lead to more conjugate gradient steps at a reduced cost per step and memory requirement.

It has been recognised that, in general, larger linear systems present

[1]a work unit in this paper is defined as the time for one step of the explicit finite difference method

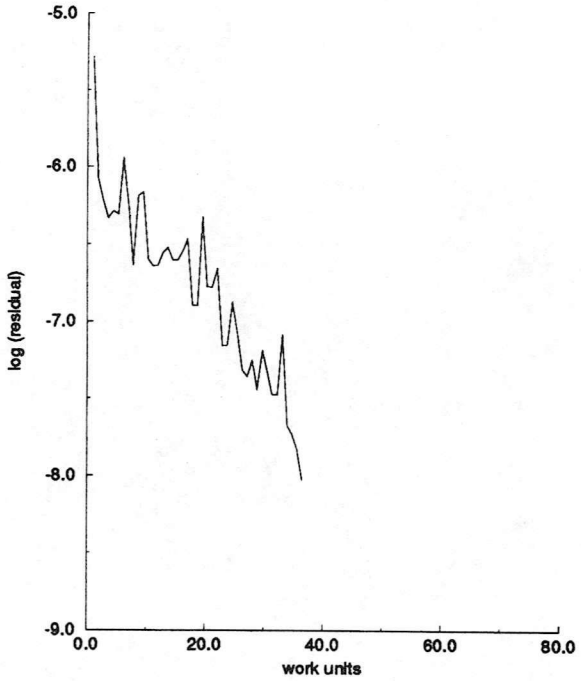


Figure 3: *The convergence for CGSTAB with ILU0 for a linear system arising from Newton's method.*

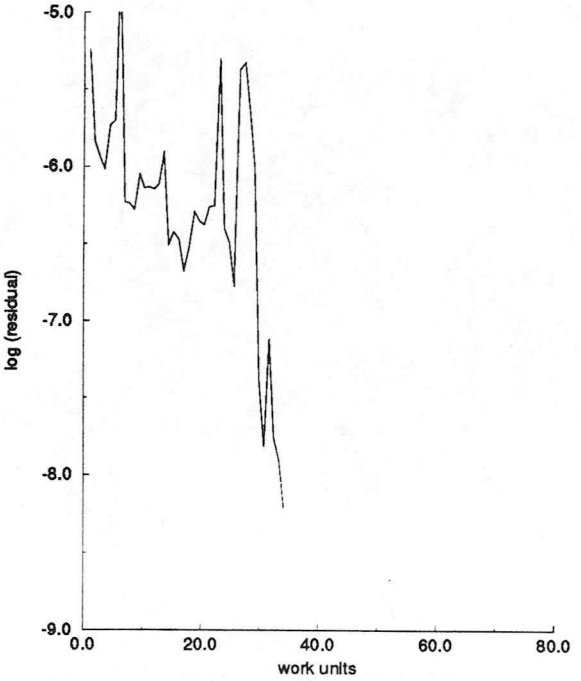


Figure 4: *The convergence for CGS with ILU0 for a linear system arising from Newton's method.*

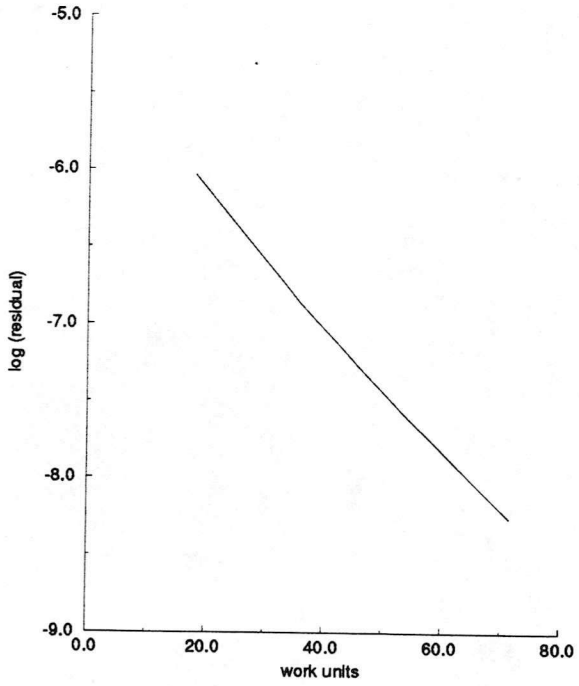


Figure 5: *The convergence for GMRES with ILU0 for a linear system arising from Newton's method.*

more difficulties for conjugate gradient methods. This means that higher levels of fill-in are required when solving systems arising from CFD applications on finer meshes. The results from the present study are summarised in table 2. There are two main conclusions to be drawn:

- on the 128 by 32 and 64 by 16 meshes ILU0 is sufficient to achieve a solution of the linear system quickly
- on finer meshes higher levels of fill-in are required at a rapidly increasing memory cost.

The published applications of Newton's method to aerofoil flows have all

mesh	preconditioner	BICG steps	entries in ILU
64 by 16	ILU0	73	72320
128 by 32	ILU0	367	296192
192 by 48	ILU0	NC	671616
192 by 48	ILU1	NC	1240848
192 by 48	ILU2	75	2071296

Table 2: *Convergence of the linear solver for various mesh resolutions. The conjugate gradient method that was used was BICG. Here, NC denotes that the solution did not converge.*

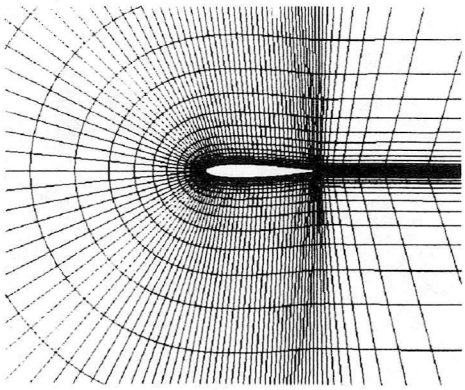


Figure 6: *NACA0012 grid with resolution 128 × 32.*

involved the damped method a term arising from the time derivative being added to the diagonal. This term not only makes the method more robust from the point of view of the starting solution but also makes the linear system easier to solve. However, the loss of quadratic convergence is disappointing when considering the overall attractiveness of the method.

5 Results

In this section we present results to demonstrate the speed up that is achieved by using Newton's method and also the accuracy that can be attained on coarse meshes using Osher's method. The two test problems used are laminar flow over a NACA0012 aerofoil at a freestream Mach number of 0.5. The first case is at zero angle of attack and the second is at 3^o. Both cases feature separated flow at the trailing edge and are standard test problems for laminar Navier-Stokes algorithms. All of the computations presented in this section were performed on an IBM RS6000/320H workstation. The mesh generation was done using the Eagle elliptic system [20] which proved to be very satisfactory. The mesh used for the two cases is shown in figure 6. A spacing of 5.0e-4 was used next to the aerofoil and the far field was located 15 chord lengths away.

The convergence for case one is shown in figure 7 and shows that a considerable speed up is achieved by using Newton's method. The flat part of the convergence graph for the explicit initialisation confirms that

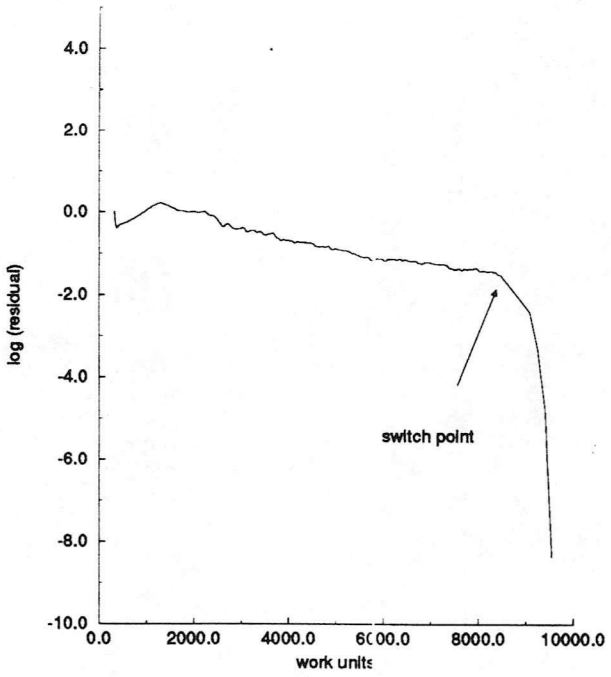


Figure 7: *Convergence for flow over a NACA0012 aerofoil at freestream Mach number of 0.5 and zero incidence. The switch point between the explicit method and Newton's method is indicated by the arrow.*

an efficient method for generating the starting solution is essential to the competitiveness of the overall method. Similar behaviour is observed in figure 8 for case two.

The pressure distribution for case one is shown in figure 9 and plots of the pressure and density contours are shown in figures 10 and 11. The separated region is evident and separation occurs at 82 percent of the chord length. This value is in excellent agreement with previous results. The comparison of the aerodynamic coefficients with previous results is shown in table 3. The present results are in excellent agreement and it is noted that the results obtained by Osher's scheme and by Roe's scheme in [6] on meshes with resolution 128×32 are comparable with results obtained on finer meshes by other methods.

The pressure distribution for case two is shown in figure 12 and plots of the pressure and density contours are shown in figures 13 and 14. The separated region is again evident and the separation point on the upper surface is at 46 percent of the chord length which is in excellent agreement with previous results. The comparison of the aerodynamic coefficients is shown in table 4 and satisfactory agreement is observed.

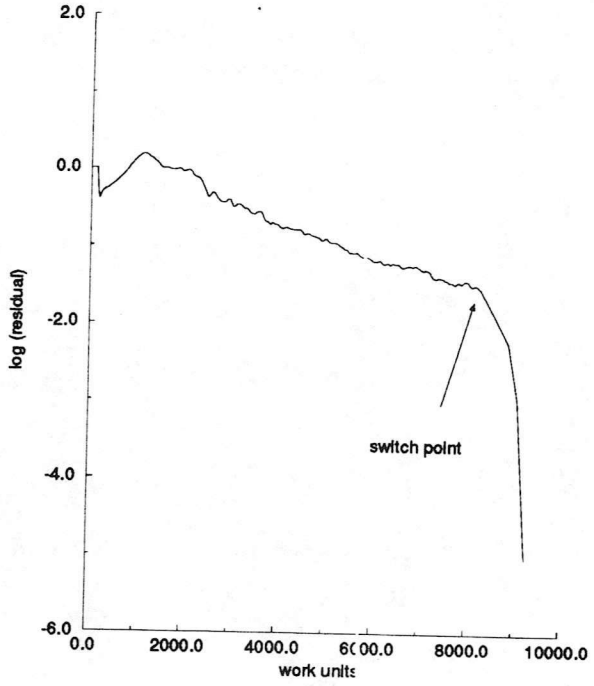


Figure 8: *Convergence for flow over a NACA0012 aerofoil at freestream Mach number of 0.5 and an incidence of 3°. The switch point between the explicit method and Newton's method is indicated by the arrow.*

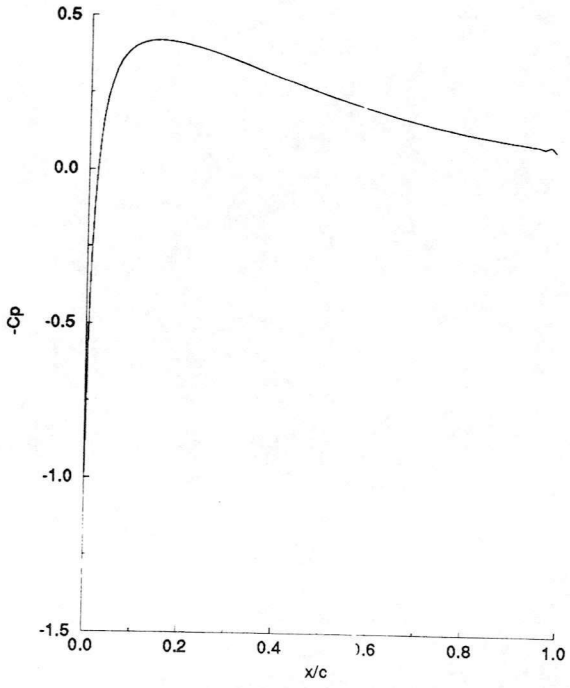


Figure 9: *Pressure distribution for flow over a NACA0012 aerofoil at freestream Mach number of 0.5 and zero incidence.*

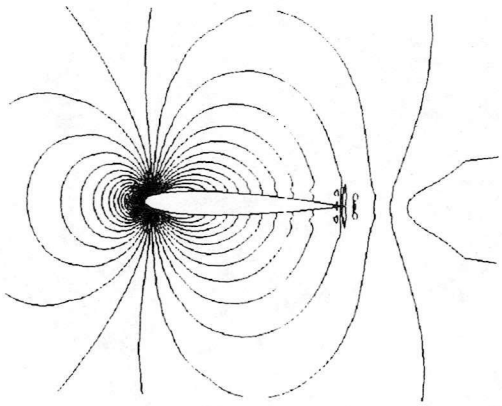


Figure 10: *Pressure contours for flow over a NACA0012 aerofoil at freestream Mach number of 0.5 and zero incidence.*

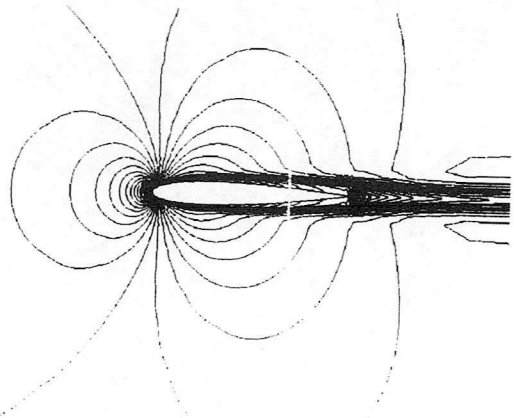


Figure 11: *Density contours for flow over a NACA0012 aerofoil at freestream Mach number of 0.5 and zero incidence.*

Author	grid	C_D^I	C_D^V
present	128×32	0.0227	0.0332
Morton et al [21]	193×49	0.0226	0.0322
Swanson and Turkel [22]	512×128	0.0224	0.0330
Venkat [6]	128×32	0.0230	0.0325
Venkat [6]	256×64	0.0225	0.0329

Table 3: *Comparison of aerodynamic coefficients for Mach 0.5 flow over a NACA0012 aerofoil at zero incidence.*

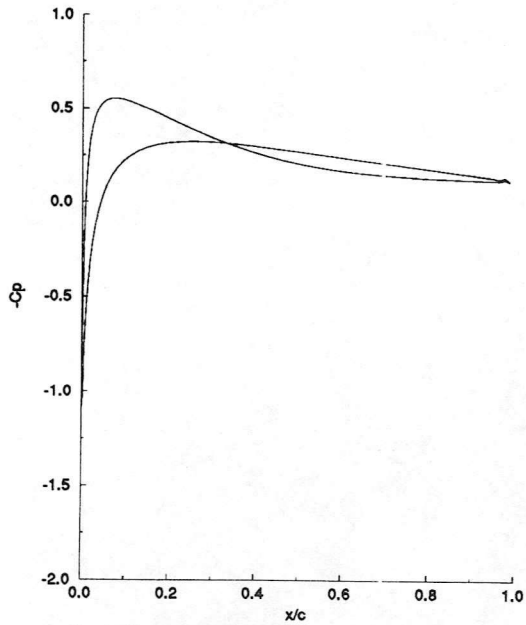


Figure 12: *Pressure distribution for flow over a NACA0012 aerofoil at freestream Mach number of 0.5 and an incidence of 3°.*

Author	grid	C_D^I	C_D^V	C_L
present	128×32	0.0237	0.0327	0.0482
Swanson and Turkel [22]	512×128	0.0263	0.0327	0.0454
Venkat [6]	128×32	0.0262	0.0321	0.0464

Table 4: *Comparison of aerodynamic coefficients for Mach 0.5 flow over a NACA0012 aerofoil at* $\alpha = 3.0°$.

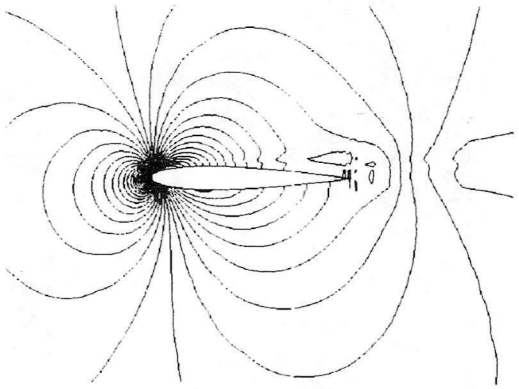


Figure 13: *Pressure contours for flow over a NACA0012 aerofoil at freestream Mach number of 0.5 and an incidence of 3°.*

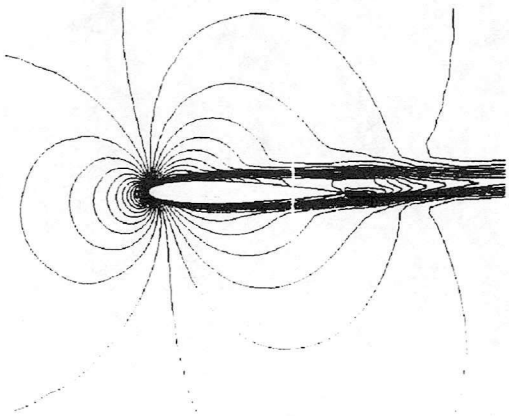


Figure 14: *Density contours for flow over a NACA0012 aerofoil at freestream Mach number of 0.5 and an incidence of 3°.*

6 Conclusions

Newton's method provides a competitive way of solving the nonlinear algebraic equations which arise from CFD discretisations if

1. a good starting for Newton's method is known
2. the required Jacobians can be evaluated
3. the linear system at each step can be solved quickly.

In this report, issues (2) and (3) have been considered.

In section 3 a method was proposed which made use of easily obtainable analytical expressions to considerably speed up the calculation of the Jacobian. Osher's flux function was differentiated numerically because of the difficulties of deriving and encoding analytical expressions. The MUSCL interpolation was differentiated analytically and the chain rule was used to provide the necessary derivatives for the Jacobian. This approach made the Jacobian calculation ten times faster than the equivalent fully numerical approach. With the increasing power of symbolic computation it is becoming simpler to perform complicated algebraic calculations and output results to a high level computer code in an efficient format. It seems likely that this will totally remove the need to derive any derivatives numerically for Newton's method.

The solution of the linear system provides the main obstacle for successful applications. Four conjugate gradient type algorithms were compared in section 4 with little advantage becoming apparent for any single one of the methods. However, preconditioning was found to be crucial to success. The level of preconditioning required was found to be dependent on the mesh size used. Storage problems were encountered on the finest mesh.

The problems considered in this report were laminar. This effectively restricted the flows to subsonic as well due to the lack of a transonic steady laminar test case. The extension to turbulent flows needs careful consideration. Various turbulent test cases have been tackled successfully using the Baldwin-Lomax model and the Johnson-King model. These are unsuitable for use with Newton's method due to their lack of differentiability and the large stencil that they involve. A more promising approach would be to use the $\kappa - \epsilon$ model.

In summary, Newton's method is an efficient way of accelerating convergence when a sufficiently good preconditioner can be calculated. Serious storage problems can be encountered on fine meshes. It should be noted that each Newton step is costly and so the quadratic convergence has to be retained to limit the number of iterations required. The cost of each iteration means that implicit solvers for unsteady problems are unlikely to

be competitive without some simplification of the method or the solution of the linear system. Methods for unsteady flow problems will be discussed in future reports.

A Symbolic Calculation of MUSCL derivatives

The symbolic package *REDUCE* was used to calculate the derivatives of the MUSCL interpolation for the Jacobian calculation and and its code optimisation package *GENTRAN* was used to generate efficient code. This is done by extracting common expressions. The *REDUCE* code used was of the form

```
% reduce code
% to calculate derivatives of MUSCL interpolation
% with Von Albada limiter
%
dl2:=u3-u2$
dl1:=u2-u1$
sl:=(2.0*dl1*dl2+epsr)/(dl1*dl1+dl2*dl2+epsr);
dell:=(1.0-xk*sl)*dl1+(1.0+xk*sl)*dl2;
ul:=u2+0.25*sl*dell;

dr2:=u4-u3$
dr1:=u3-u2$
sr:=(2.0*dr1*dr2+epsr)/(dr1*dr1+dr2*dr2+epsr);
delr:=(1.0+xk*sr)*dr1+(1.0-xk*sr)*dr2;
ur:=u3-0.25*sr*delr;

optimize jacl(1):=:df(ul,u1),
jacl(2):=:df(ul,u2),
jacl(3):=:df(ul,u3),
jacr(1):=:df(ur,u2),
jacr(2):=:df(ur,u3),
jacr(3):=:df(ur,u4) } iname a;
```

and the Fortran code that is produced is

```
% fortran code
%
%
A3=U3-U2
A4=U2-U1
A65=A3*A3+EPSR
A5=A65+A4*A4
A46=A5*A5
```

```
A38=U2*XK
                    .
                    .
                    .
JACL(1)=(A98*A89-(A5*(A71*(2.0*A61+A59+A58+3.0*A47-(4.0*A41)-
. (8.0*
. A37)+A48)+A98*A91+A97*A85)))/A6
A30=U3-(2.0*U2)+U1
A94=2.0*U3*U2
A60=A94+A47*XK
A95=6.0*A37
JACL(2)=(A5*(2.0*A46+A30*(A85-A91)+A71*(A61+A60+A59+A47-
. (6.0*A41)-
. A48-A95))+A89*A30)/(2.0*A53)
JACL(3)=(A5*(A71*(A58+A47+8.0*A41+4.0*A37+3.0*A48-(2.0*A60)-
. A59)+
. A97*A91+A98*A85)-(A97*A89))/A6
A17=U3-U4
A18=A65+A17*A17
                    .
                    .
                    .
JACR(1)=(A18*(A96*A84-(A97*A90)-(A70*(A77+A3*(A97-A93)-
. (A17*(A97+
. A93)))))-(A97*A88))/A19
A64=U4-(2.0*U3)
A29=-A64-U2
JACR(2)=(A88*A29-(A18*(A29*(A84-A90)+A70*(A94-A42-A56+A32+
. 6.0*A33+
. A64*U4+U4*XK*(4.0*U2+U4-(6.0*U3))-A95)-(2.0*A50))))/(2.0*A52)
A92=2.0*XK*A3
JACR(3)=(A18*(A70*(A76+A3*(A92-A96)+A17*(A96+A92))+A97*A84-
. (A96*
. A90))-(A96*A88))/A19
```

References

[1] N.Qin and B.E.Richards. Sparse quasi-Newton method for high resolution schemes. *Notes in Numerical Fluid Dynamics*, 20:310–317, 1988.

[2] N.Qin and B.E.Richards. Sparse quasi-Newton method for navier-stokes solution. *Notes in Numerical Fluid Dynamics*, 29:474–483, 1990.

[3] N.Qin X.Xu and B.E.Richards. A new parallelizable iterative solver for large sparse nonsymmetric linear systems arising from CFD. *Int. J. Num. Mths. Fluids*, 15:613–623, 1992.

[4] M.J.D.Powell A.Curtis and J.K.Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Maths. Applics*, 13:117–120, 1974.

[5] V.Venkatakrishnan. Newton solution of inviscid and viscous problems. *A.I.A.A. Journal*, 27(7):885–891, 1989.

[6] V.Venkatakrishnan. Viscous computations using a direct solver. *Computers and Fluids*, 18(2):191–204, 1990.

[7] V.Venkatakrishnan. Preconditioned conjugate gradient methods for the compressible Navier-Stokes equations. *A.I.A.A. J.*, 29:1092–1100, 1990.

[8] D.L.Whitfield and L.K.Taylor. Discretized Newton-relaxation solution of high resolution flux-differnce split schemes. Technical report, A.I.A.A. 91-1539-CP, 1991.

[9] P.D.Orkwis and D.S.McRae. Newton's method solver for high-speed viscous separated flowfields. *A.I.A.A. J.*, 30:78–85, 1991.

[10] M.Hafez C.P.Van Dam and J.Ahmad. Calculation of viscous flows with separation using Newton's method and direct solver. Technical report, A.I.A.A. 88-0412-CP, 1988.

[11] A.B.Stephens G.R.Shubins and H.M.Glaz. Steady shock tracking and Newton's method applied to one-dimensional duct flow. *J. Comp. Phys.*, 39:364–374, 1980.

[12] S.Osher and S.R.Chakravarthy. Upwind schemes and boundary conditions with applications to Euler equations in general coordinates. *J. Comp. Phys.*, 50:447–481, 1983.

[13] B.Van Leer. Towards the ultimate conservative difference scheme. V:a second–order sequel to Godunov's method. *J. Comput. Phys.*, 32:101–136, 1979.

[14] B.Van Leer G.D.Van Albada and W.W. Roberts. A comparative study of computational methods in cosmic gas dynamics. *Astronomy and Astrophysics*, 108, 1982.

[15] R.Fletcher. Proceedings of conference in numerical analysis. *Lecture Notes in Mathematics*, 506:73–98, 1976.

[16] H.A.Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Stat. Comp.*, 13:631–644, 1992.

[17] P.Sonneveld. CGS: A fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Stat. Comp.*, 10:36–52, 1989.

[18] Y.Saad and M.H.Schultz. GMRES: A general minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Stat. Comp.*, 7:856–869, 1986.

[19] J.A.Meijerink and H.A.Van der Vorst. An iterative solution method for linear equations of which the coefficient matrix is a symmetric M matrix. *Math Comp*, 31:148–162, 1977.

[20] J.F.Thompson L.E.Lijewski, J.Cipolla and B.Gatlin. Program Eagle user's manual, vol I - introduction and grid applications. Technical Report 117, A.F.A.T.L.-TR-88, 1988.

[21] J.A.McKenzie P.I.Crumpton and K.W.Morton. Cell vertex algorithms for the compressible Navier-Stokes equations. Numerical Analysis Report 12, Oxford University Computing Laboratory, 1991.

[22] R.C.Swanson and E.Turkel. Artificial dissipation and central difference schemes for the Euler and Navier-Stokes equations. Technical report, A.I.A.A. 87-1107-CP, 1987.